allinea

Leaders in parallel software development tools

Debugging CUDA Applications with Allinea DDT

Ian Lumb

Sr. Systems Engineer, Allinea Software Inc.

ilumb@allinea.com

GTC 2013, San Jose, March 20, 2013

Challenges for Developers

Heterogeneity

Scale

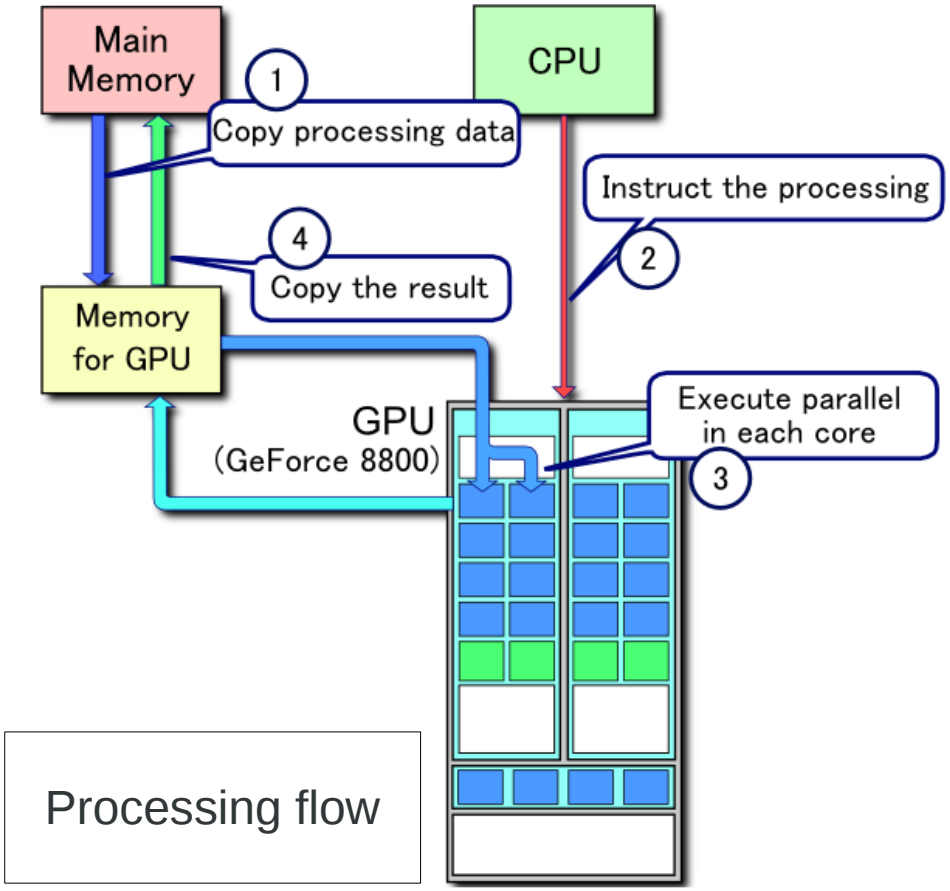
Legacy

Diversity

allinea
www.allinea.com

Embracing GPUs

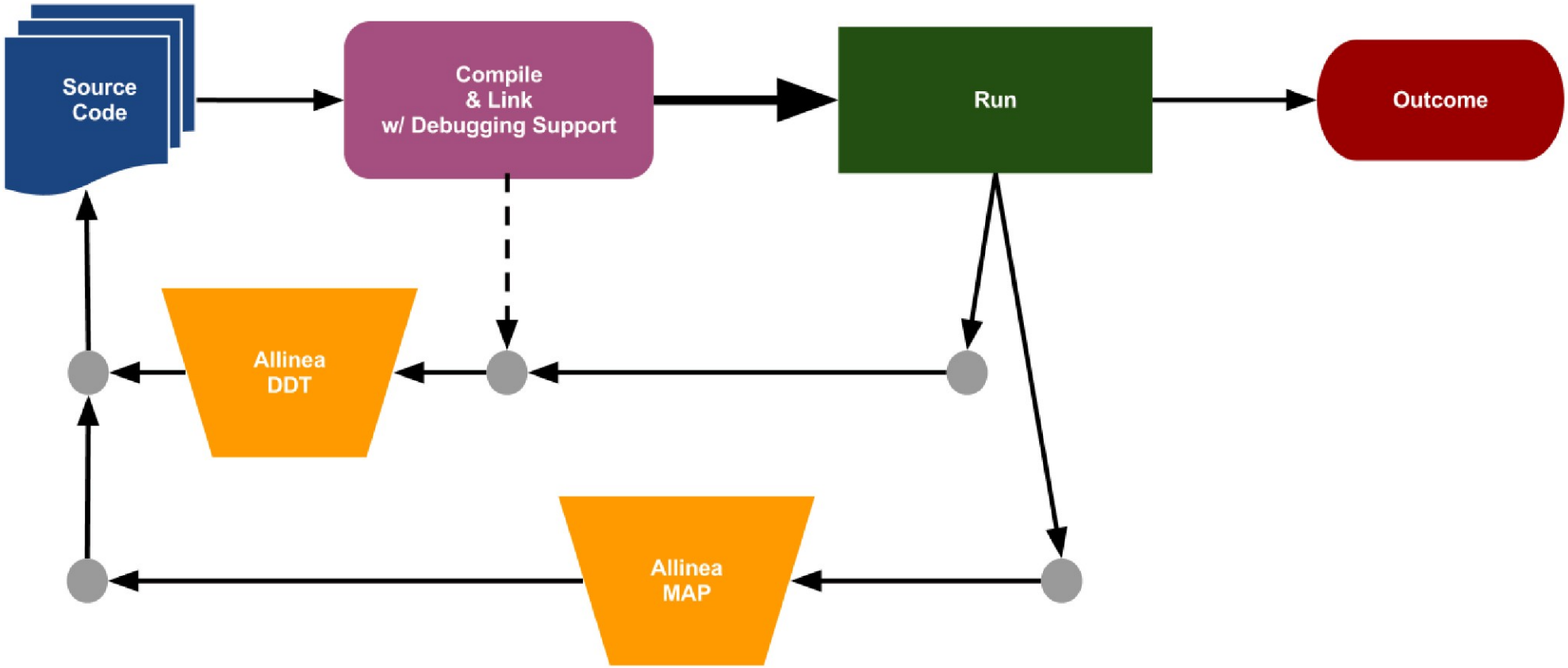
- GPUs – a rival to traditional processors
  - Great price/performance ratios
- New languages, compilers, standards
  - CUDA, OpenACC, OpenCL, ...
- HPC developers need to consider
  - Data transfer
  - Multiple memory levels
  - Grid/block layout and thread scheduling
  - Synchronization
- Bugs are **inevitable**



The Allinea Environment: Benefits

- **At last:** a modern **integrated** environment for the HPC developer
- Supporting the lifecycle of application development and improvement
  - Productively debug code
  - Enhance application performance
- Designed for productivity
  - Consistent integrated easy to use tools
  - Enables effective HPC development
- Improve system usage
  - Fewer failed jobs
  - Higher application performance

allinea
www.allinea.com


Source
Code
Compile
& Link
w/ Debugging Support
Run
Outcome
Allinea
DDT
Allinea
MAP

allinea

Leaders in parallel software development tools

allinea DDT

- Full-scale parallel debugging
- Full CUDA 5 support
- OpenMP + OpenACC
- Mixed MPI / OpenMP / CUDA
- Debug the real device

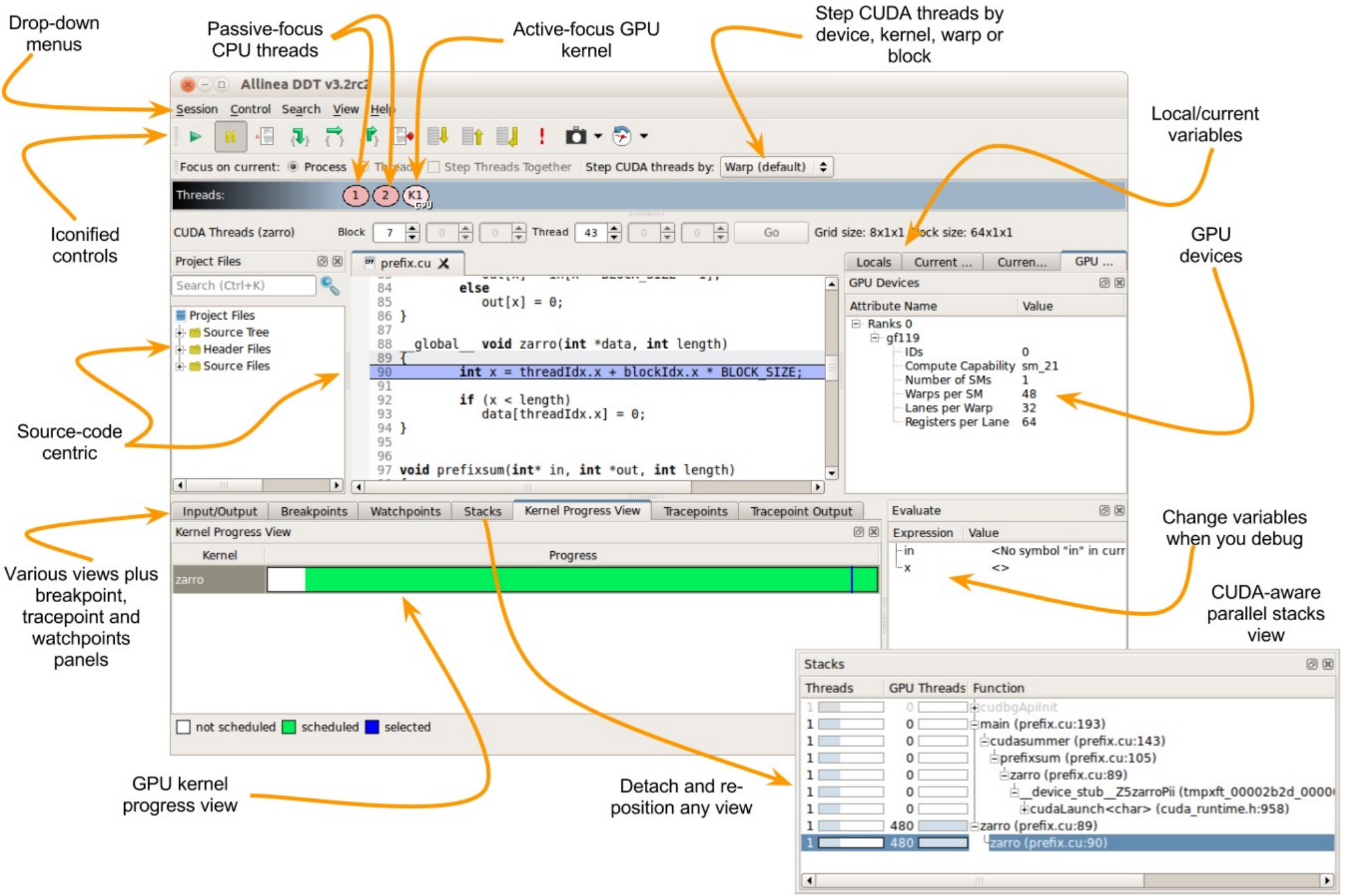
World-class Parallel Debugging

www.allinea.com

allinea

Leaders in parallel software development tools

Allinea DDT Demo:
CUDA Debugging Essentials


Drop-down menus
Passive-focus CPU threads
Active-focus GPU kernel
Step CUDA threads by device, kernel, warp or block
Local/current variables
Iconified controls
GPU devices
Source-code centric
Change variables when you debug
Various views plus breakpoint, tracepoint and watchpoints panels
CUDA-aware parallel stacks view
GPU kernel progress view
Detach and re-position any view
Allinea DDT v3.2rc2
Session Control Search View Help
Focus on current: Process Thread Step Threads Together Step CUDA threads by: Warp (default)
Threads: 1 2 K1
CUDA Threads (zarro) Block 7 Thread 43 Go Grid size: 8x1x1 Block size: 64x1x1
Project Files
Search (Ctrl+K)
Project Files
Source Tree
Header Files
Source Files
prefix.cu
else
out[x] = 0;
}
__global__ void zarro(int *data, int length)
{
int x = threadIdx.x + blockIdx.x * BLOCK_SIZE;
if (x < length)
data[threadIdx.x] = 0;
}
void prefixsum(int* in, int *out, int length)
Locals Current ... Curren... GPU ...
GPU Devices
Attribute Name Value
Ranks 0
gf119
IDs 0
Compute Capability sm_21
Number of SMs 1
Warps per SM 48
Lanes per Warp 32
Registers per Lane 64
Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output
Kernel Progress View
Kernel Progress
zarro
not scheduled scheduled selected
Evaluate
Expression Value
in <No symbol "in" in curr
x <>
Stacks
Threads GPU Threads Function
cudbgApiInit
main (prefix.cu:193)
cudasummer (prefix.cu:143)
prefixsum (prefix.cu:105)
zarro (prefix.cu:89)
__device_stub__Z5zarroPii (tmpxft_00002b2d_0000
cudaLaunch<char> (cuda_runtime.h:958)
zarro (prefix.cu:89)
zarro (prefix.cu:90)

Debugging Dynamic Parallelism

```
__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}
```

Allinea DDT v4-0-BRANCH [Trial Version]

File View Control Search Tools Window Help

Focus on current: Process Thread Step Threads Together Step CUDA threads by: Warp (default)

Threads: 1 2 K4 K5 K6 K7

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Go Grid size: 1x1x1 Block size: 256x1x1

Project Files

Search (Ctrl+K)

- stddef.h
- stdint.h
- stdio.h
- stdlib.h
- str-two-way.h
- string2.h
- surface_types.h
- sysmacros.h
- texture_types.h
- thread_db.h
- thread_dbPh
- time.h
- tls.h
- tlsdeschtab.h
- tmmintrin.h
- types.h
- unistd.h
- unwind.h
- utmp-equal.h
- vector_types.h

permute.cu | vector_types.h

```
1  #include <stdio.h>
2
3  __global__ void permute(int n, int *data) {
4      __shared__ int smem[256];
5      if (n <= 1 || threadIdx.x >= n)
6          return;
7      smem[threadIdx.x] = data[threadIdx.x];
8      __syncthreads();
9      smem[threadIdx.x] *= 2; // permute data in a trivial way
10     __syncthreads();
11     // Write back to GMEM since we can't pass SMEM to children.
12     data[threadIdx.x] = smem[threadIdx.x];
13     __syncthreads();
14     if (threadIdx.x == 0) {
15         permute<<< 1, 256 >>>(n/2, data);
16         permute<<< 1, 256 >>>(n/2, data+n/2);
17         cudaDeviceSynchronize();
18     }
19 }
20
21 int main(int argc, char** argv) {
```

Locals | Current Line(s) | Current Stack

Current Line(s)

Variable Name	Value

Type: none selected

Input/Output* | Breakpoints | Watchpoints | Stacks | Kernel Progress View | Tracepoints | Tracepoint Output

Stacks

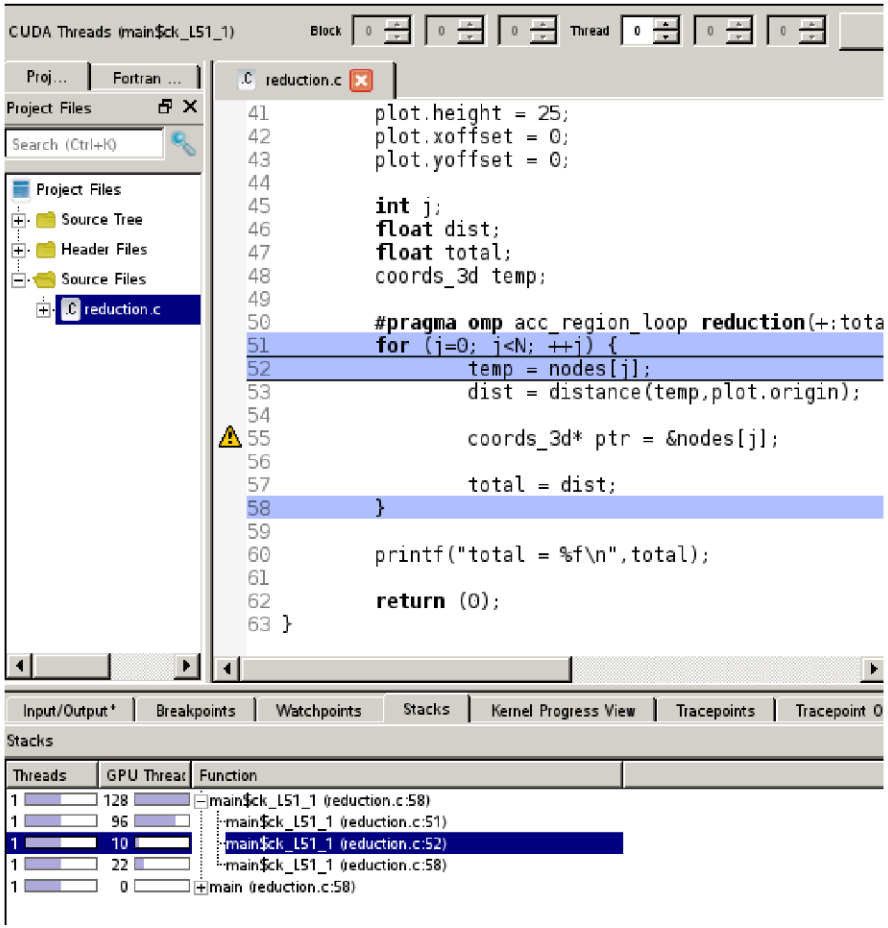
Threads	Cuda Threads	Function
4	128	permute (permute.cu:3)
1	32	permute (permute.cu:12)
2	2	permute (permute.cu:16)
1	1	permute (permute.cu:17)
3	93	permute (permute.cu:19)
1	0	main (permute.cu:32)
1	0	cudbgApiInit

Evaluate

Expression	Value
data	
output[42]	<No symbol "output" in current context.>

Ready

OpenACC Debugging



- On device debugging with Allinea DDT
  - Variables – arrays, pointers, full F90 and C support
  - Set breakpoints and step warps and blocks
- Identical to CUDA
  - Full warp/block/kernel controls
  - CPU vs GPU distinction still present
- Status
  - Allinea supports on-device debugging for Cray compiler
  - Other compilers to follow

allinea
www.allinea.com

allinea

Leaders in parallel software development tools

- Full-scale parallel profiling
- Just 5% runtime overhead
- No need to instrument code
- MPI, memory and CPU
- See the slowest lines of code

allinea MAP

World-class Parallel Profiling

www.allinea.com

Use of Allinea DDT at SciNet

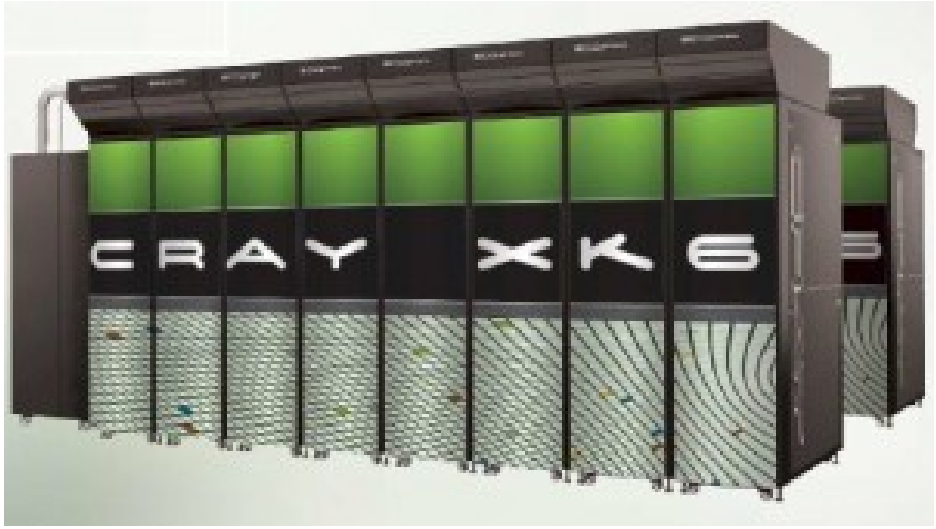
- Productively **debug** parallel CUDA code
- Completely **understand** parallel CUDA code
  - Interact with data, algorithms, codes, programs and applications in real time
- **Develop** parallel CUDA code from scratch
- **Port** parallel algorithms, codes, programs and applications to CUDA
- **Scale** CUDA algorithms, codes, programs and applications

http://www.scinethpc.ca/

allinea
www.allinea.com

Petascale Debugging

- Cray XK* systems
  - ORNL Titan
    - MPI debugging proven at 220,000 CPU cores in 2010
      - ~300,000 CPU cores in 2012
    - MPI-OpenACC hybrid codes *expected* to scale similarly
  - NCSA Blue Waters
    - ~700,000 CPU cores



Allinea at GTC 2013

- Please visit our booth!!!
  - Live demonstrations
  - Further information

- Call to action: Evaluate Allinea tools!

  http://www.allinea.com/products/

allinea
www.allinea.com

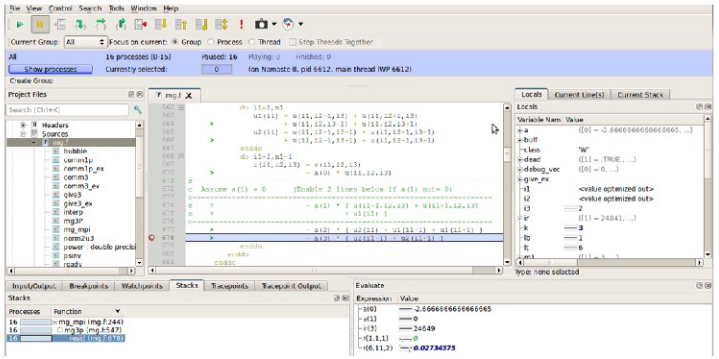
allinea

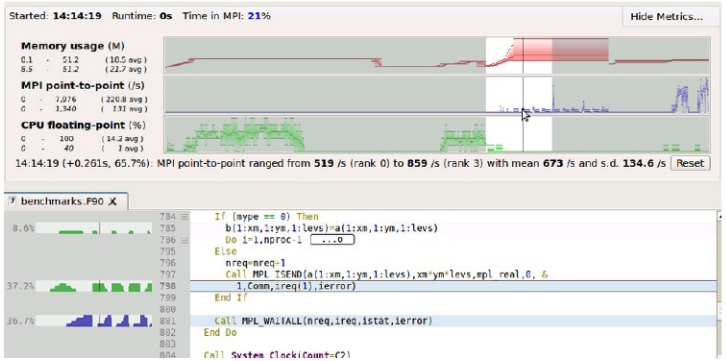
Leaders in parallel software development tools

The Allinea Environment - an integrated, **ready-to-run** development suite

Get **correct** results fast using the industry-leading parallel debugger

Full support for CUDA 5





See which loops can be offloaded to the GPU most **effectively** with Allinea MAP

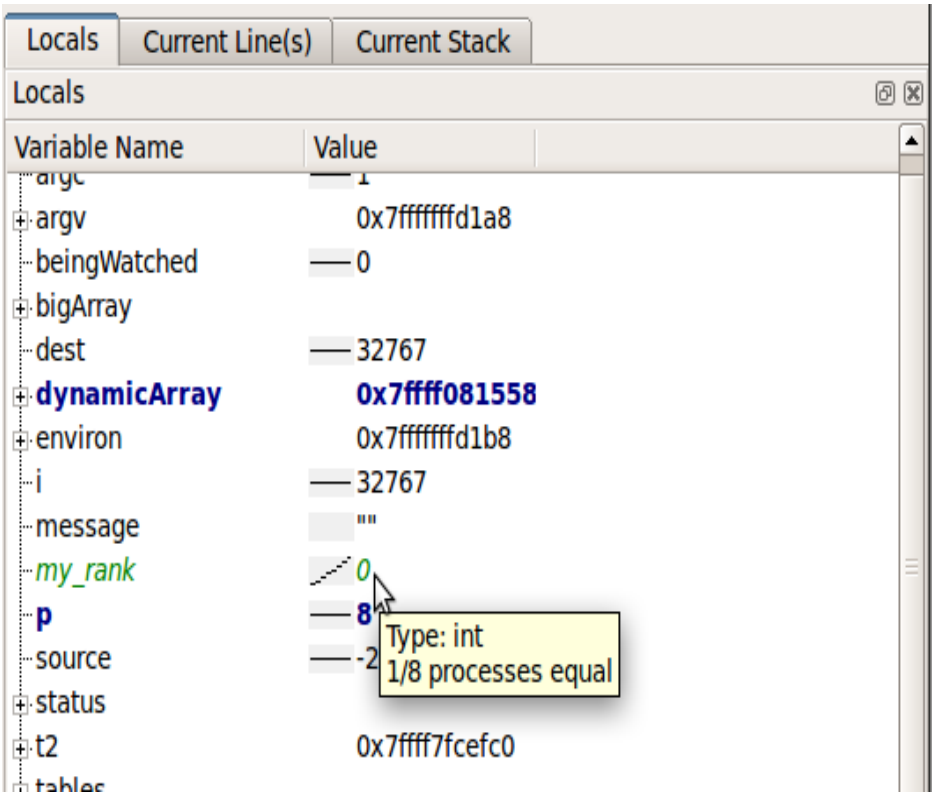
allinea

Leaders in parallel software development tools

www.allinea.com

Allinea DDT: Fixing bugs made easy

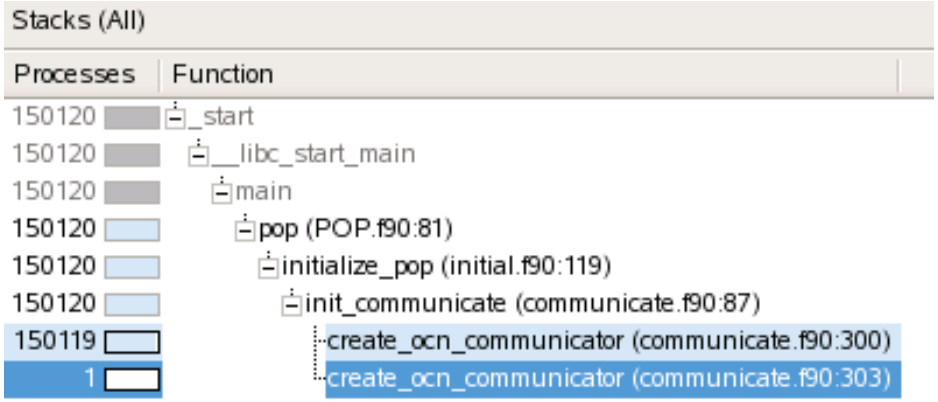
- A tool that allows you to solve your problems faster
  - Control threads and processes en-masse
  - Syntax-highlighting source browser
  - Parallel stacks and variable views that highlight divergence
- Supports the latest in MPI, OpenMP, CUDA and more
  - CUDA 5.0 and Kepler K20
  - Intel Xeon Phi coprocessor

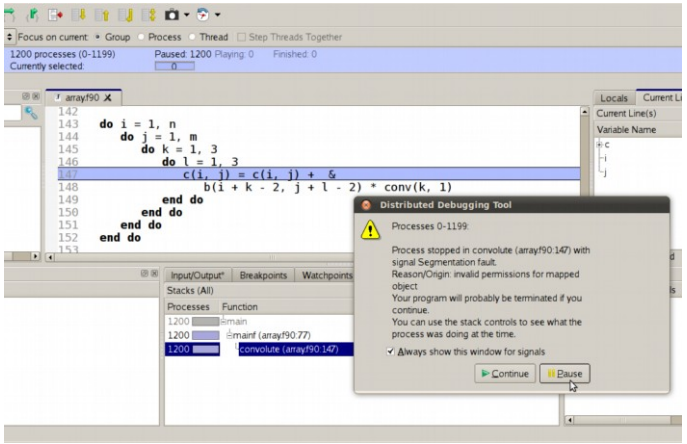


allinea
www.allinea.com

Allinea DDT: Proven to the extreme

- Scalability by design
  - User interface that scales
  - High performance tree architecture
- Proven performance at Petascale
  - Measured in milliseconds
  - Routine use at 100,000+ cores





allinea
www.allinea.com

Allinea DDT: More than debugger

- Integrated automated detection of bugs
  - Static analysis
  - Memory leaks and errors
- Open plugin architecture
  - MPI checking tools
- Offline mode - debug in batch mode

```
30      threads = calloc(sizeof(pthread_t), nthreads);
31      ids = calloc(sizeof(int), nthreads);
32
33      init_mutex();
34
35      pthread_mutex_lock(mutley);
36      for (i = 0; i < nthreads; ++i) {
37          ids[i] = i;
38          pthread_create (threads + i, NULL, &thread,
39      }
40      pthread_mutex_unlock(mutley);
41      for (i = 0; i < nthreads; ++i)
42          pthread_join (threads[i], NULL);
43
44      return 0;
45  }
46  ...oid *q)
47  {
49      volatile int busy = 0;
50      volatile int locker = 0;    /* to be amended by
51      int i, j;
52      double k = 1;
53      int tid = *(int*) q;
54
55      usleep(rand() % 31);
56
```

error Memory leak: threads

error Memory leak: ids

allinea
www.allinea.com

allinea

Leaders in parallel software development tools

www.allinea.com

Allinea DDT 3.2.1 (10/2012)

- CUDA 5.0 support
- Intel Xeon Phi support
- MPMD support
  - MPICH 2
  - Intel MPI
- IBM BlueGene /Q port

Allinea Tools 4.0 (3/2013)

- GA introduction of Allinea MAP
  - Observe and improve the performance of your MPI application
- Improved syntax highlighting and code folding in code viewer
- Remote client
  - Native client for Linux, Windows and Mac
- Redesigned welcome page
- Thread viewer
  - Shows OpenMP thread ID for supported implementations

Allinea Tools 4.0 (2)

- Support for MPICH 3
- Cray-specific
  - Attach to GPU codes (Cray XK7)
  - Memory Debugging Library preloading
    - C/Fortran (no threads/threads) now supported (Cray XK6/7)
    - C++ (no threads/threads) not supported

allinea

Leaders in parallel software development tools

www.allinea.com

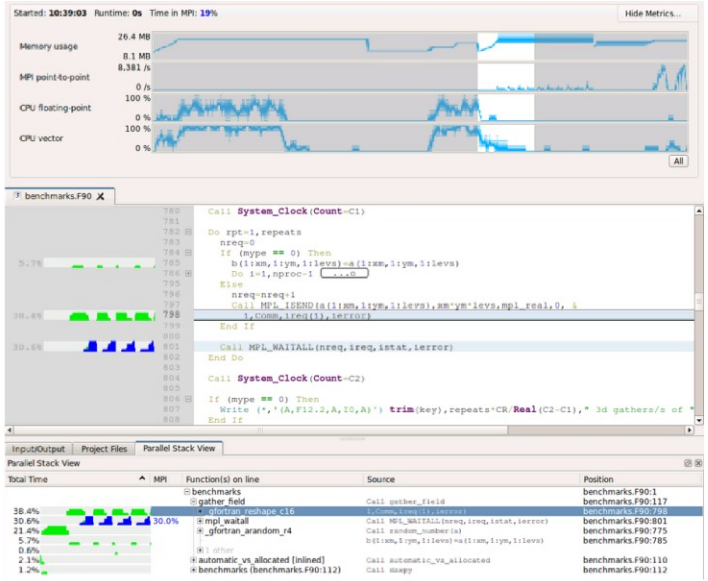
allinea

Leaders in parallel software development tools

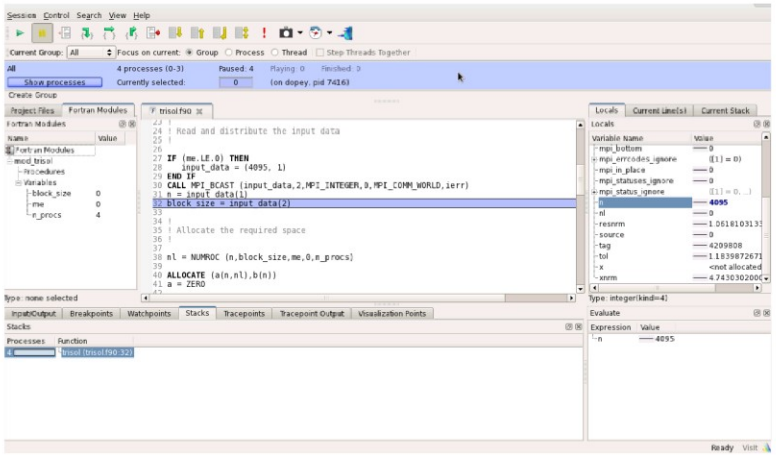
www.allinea.com

A Unified Environment for HPC

Shared Graphical Interface



Shared Configuration Files



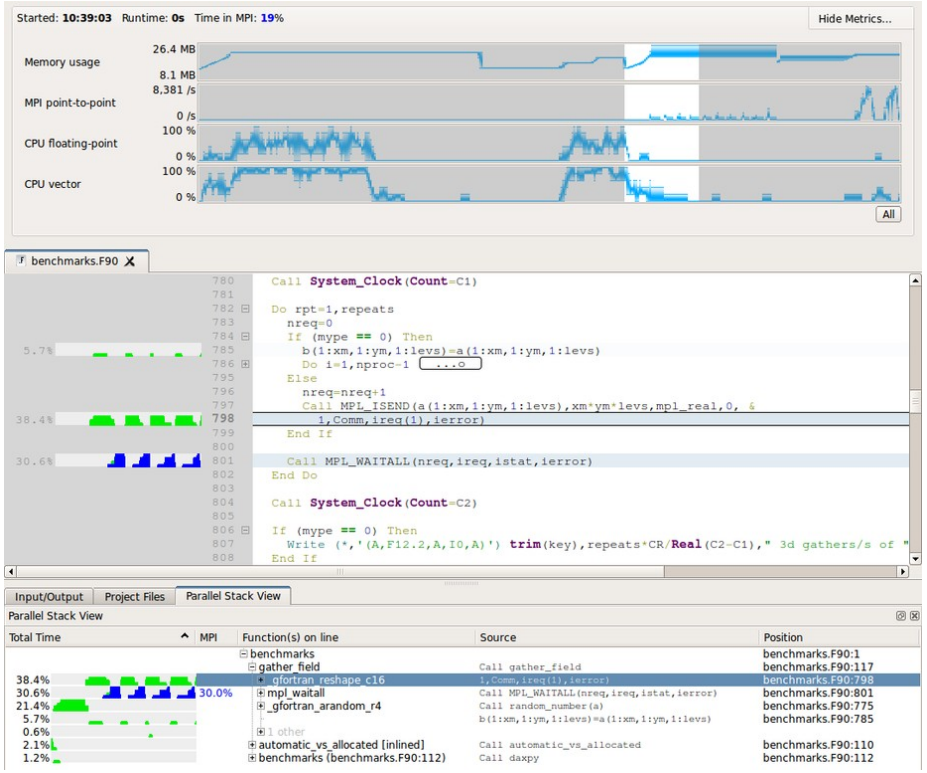
Shared Scalable Architecture

Intelligent data consolidation

allinea
www.allinea.com

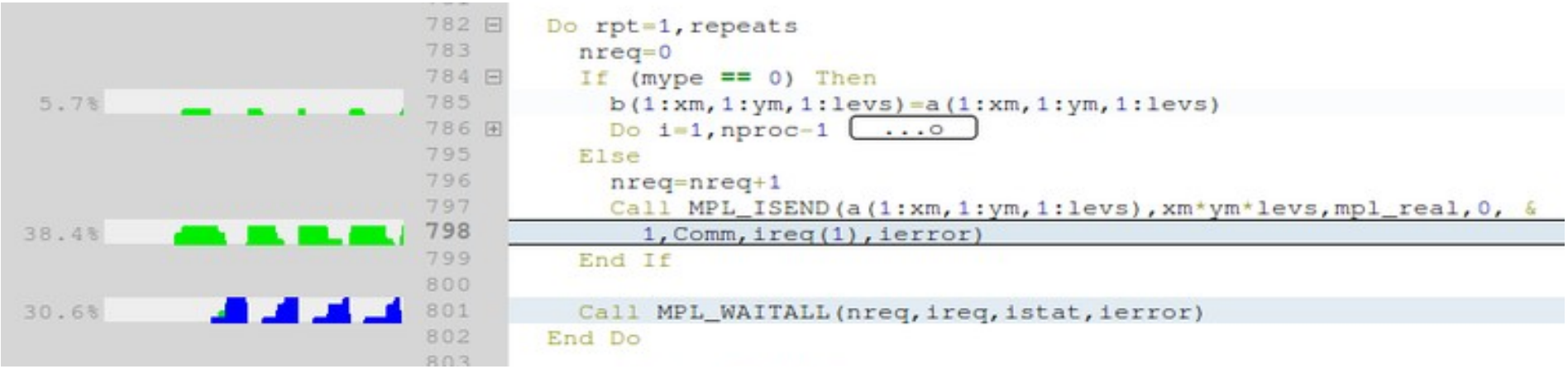
Allinea MAP: The profiler you'll want to use

- Works first time, every time
- From one process to tens of thousands
  - 5% slowdown
  - No instrumentation and no huge data files
- No need to recompile



allinea
www.allinea.com

Allinea MAP: Refreshing simple

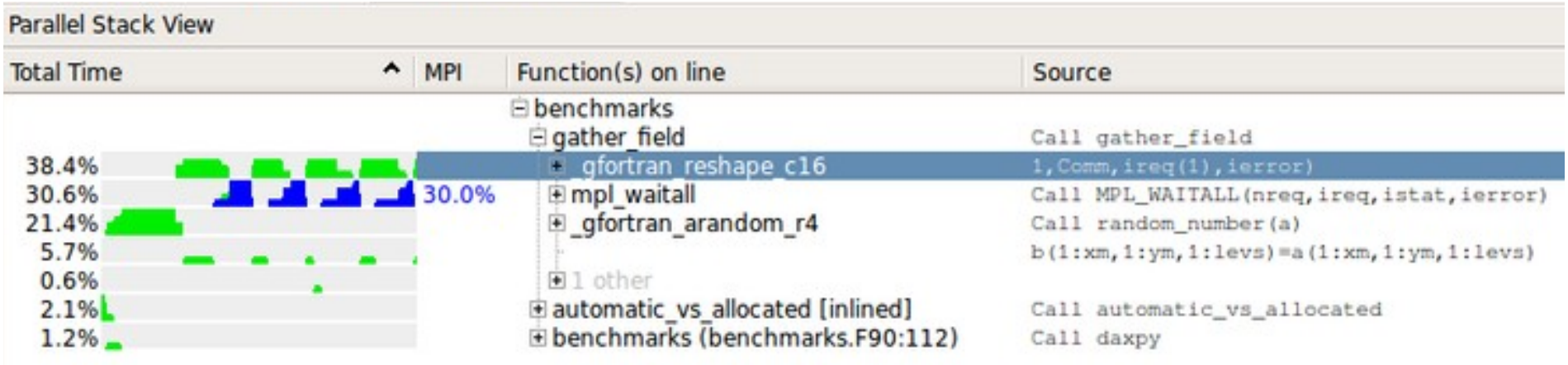


- See where time is spent in your source code.
- Visualize the entire run
- Zoom in to explore iterations, functions and loops.

Allinea MAP: Surprisingly deep

- A wide range of metrics for deep insights into performance:
  - CPU: Time spent in FPU, vector (SSE/AVX) and memory operations
  - MPI: Time spent and bytes transferred in P2P / collective operations
- Visual display of distribution over ranks, min / max ranks labelled
- Export data via XML or CSV for deeper analysis and reports

Allinea MAP: Built on strength



- World-class scalability
  - Shares Allinea DDT tree architecture – proven beyond Petascale
  - Data is merged on the cluster: no huge files.

allinea
www.allinea.com